In the Specification

Please amend the specification of this application as follows:

Rewrite the paragraph at page 24, lines 1 to 4 (Table 1) as follows:

--

Symbol	Operation
~	bit wise NOT
&	bit wise AND
|	bit wise OR
^	bit wise exclusive OR
%%	mask generation
%%!	modified mask generation
>>	shift right
||	parallel operation

Table 1--

Rewrite the paragraph at page 44, line 20 to page 45, line 9 as follows:

--The "E" bit (bit 14) designates an explicit multiple carry-in. This bit permits the carry-in to be specified at run time by the input to the C-port of arithmetic logic unit 230. If both the "A" bit and the "E" bit are "1" and the "FMOD" field does not designate the cin function, then the effects of the "S", "I" and "C" bits are annulled. The carry input to each section during multiple arithmetic is taken as the exclusive OR of the least significant bit of the corresponding section input to the C-port and the function signal F0. If multiple arithmetic is not selected the single carry-in to bit 0 of arithmetic logic unit 230 is the exclusive OR of the least significant bit (bit 0) the input to the C-port and the function signal F0. This is particularly useful for performing multiple arithmetic in which differing functions are

performed in different sections. One extended arithmetic logic unit operation corresponds to ~~(A^B)&C ³ (A^~B)&C~~ (A^B)&C | (A^~B)&C. Using a mask for the C-port input, a section with all "0's" produces addition with the proper carry-in of "0" and a section of all "1's" produces subtraction with the proper carry-in of "1".--

Rewrite the paragraph at page 48, line 12 to page 49, line 4 as follows:

--The output of multiplier 220 supplies the input of product left shifter 224. Product left shifter 224 can provide a controllable left shift of 3, 2, 1 or 0 bits. The output of multiply shift multiplexer MSmux 225 controls the amount of left shift of product left shifter 224. Multiply shift multiplexer MSmux 225 selects either bits 9-8 from the "DMS" field of data register D0 or all zeroes depending on the instruction word. In the preferred embodiment, multiply shift multiplexer MSmux 225 selects the "0" input for the instructions ~~MPYx ° ADD and MPYx ° SUB~~ MPYx ‖ ADD and MPYx ‖ SUB. These instructions combine signed or unsigned multiplication with addition or subtractions using arithmetic logical unit 230. In the preferred embodiment, multiply shift multiplexer MSmux 225 selects bits 9-8 of data register D0 for the instructions ~~MPYx ° EALUx~~ MPYx ‖ EALUx. These instructions combine signed or unsigned multiplication with one of two types of extended arithmetic logic unit instructions using arithmetic logic unit 230. The operation of data unit 110 when executing these instructions will be further described below. Product left shifter 224 discards the most significant bits shifted out and fills the least significant bits shifted in with zeros. Product left shifter 224 supplies a 32 bit output connected to a second input of multiplexer Rmux 221.--

Rewrite the paragraph at page 65, line 10 to page 66, line 10 as follows:

--Figure ~~17~~ 9 illustrates the steps typically executed when a document specified in a page description language, such as PostScript, is to be printed. Following receipt of the print file (input data file ~~301~~ 401) is interpretation (processing block ~~302~~ 402). In this step, the input PostScript file is interpreted and converted into an intermediate form called the display list (data file ~~303~~ 403). The display list ~~303~~ 403 consists of a list of low level primitives such as trapezoids, fonts, images, etc. that make up the described page. Next the display list is rendered (processing block ~~304~~ 404). Each element in the display list ~~303~~ 403 is processed in this step and the output is written into a buffer known as the page buffer (data file ~~305~~ 405). The page buffer ~~305~~ 405 represents a portion of the output image for a particular color plane. In the page buffer ~~305~~ 405, each pixel is typically represented by 8 bits. After all the elements in display list ~~303~~ 403 have been processed, page buffer ~~305~~ 405 contains the output image in an 8 bit format. Next the page buffer is screened (processing block ~~306~~ 406). The resolution supported by the printing device may be anywhere between 1 to 8 bits per pixel. Page buffer ~~305~~ 405 developed in the rendering step ~~304~~ 404 has to be converted into the resolution supported by the printer. The thus converted data is called the device image. Each pixel in page buffer ~~305~~ 405 has to be converted to its corresponding device pixel value. For instance, in the case of a 4 bit device pixel, each pixel in page buffer ~~305~~ 405 has to be converted to a 4 bit value. This process called screening results in a screened page buffer (data file ~~307~~ 407). Next comes printing (processing block ~~308~~ 408). Each pixel in the screened page buffer ~~307~~ 407 is printed on the paper. This process is repeated for all the color planes, cyan, yellow, magenta and black.--

Rewrite the paragraph at page 66, lines 14 to 16 as follows:

--~~$O_k(n) = \frac{1}{2^k - 1}\ int[\ (2^k - 1)I(n) + D(n)\ ]$~~

$$O_k(n) = \frac{1}{2^k - 1}\ int[\ (2^k - 1)I(n) + D(n)\ ]\text{--}$$

Rewrite the paragraph at page 68, lines 28 and 29 as follows:

--2. ~~Y = X & ~@MF + ¥~~ FF & @MF

(Y1:Y2:Y3:Y4) = (X1:X2:X3:X4) & ~@MF + (FF:FF:FF:FF) & @MF--